Package 'binSegInf'

November 23, 2016

Title Binary Segmentation Inference

Version 0.0.0.9000

Description Binary Segmentation Inference

Depends R (>= 3.3.1),
Matrix,
pryr,
testthat,
stringr

License What license is it under?

LazyData true

RoxygenNote 5.0.1.9000

R topics documented:

`add.cplist` *Function to add entries to cplist*

Description

Function to add entries to cplist

Usage

```
## S3 method for class 'cplist'
add(cplist, new.j, new.k, newentry)
```

Arguments

`cplist`	list containing an n x 3 matrix, and the index of the last nonempty (i.e. not all NA's) row.

`binary_search` *Binary search an integer vector for goal Returns index of goal, in the vector.*

Description

Binary search an integer vector for goal Returns index of goal, in the vector.

Usage

```
binary_search(vec, goal, verbose = FALSE)
```

`binseg.by.size` *Function to carry out fixed-number-of-steps binary segmentation.*

Description

Function to carry out fixed-number-of-steps binary segmentation.

Usage

```
binseg.by.size(y, numsteps, verbose = FALSE)
```

Arguments

`y`	numeric vector, data
`numsteps`	desired number of changepoints
`verbose`	set to true for algorithm run details.

`binseg.by.thresh` *Main function for binary segmentation for fixed threshold. This is actually a wrapper for binary segmentation with fixed threshold. It creates an environment and creates the variables there, then runs binseg.by.thresh.inner() all in this environment, and returns the relevant guy*

Description

Main function for binary segmentation for fixed threshold. This is actually a wrapper for binary segmentation with fixed threshold. It creates an environment and creates the variables there, then runs binseg.by.thresh.inner() all in this environment, and returns the relevant guy

Usage

```
binseg.by.thresh(y, thresh, s = 1, e = length(y), j = 0, k = 1,
  verbose = FALSE, return.env = FALSE)
```

Arguments

y	The original data.
thresh	Threshold for $\tilde{X}$. This serves as a stopping rule for the recursion.
s	Starting index, in the vector-valued data. Must be an integer larger than or equal to 1, and strictly smaller than e.
e	Ending index, in the vector-valued data. Must be an integer smaller than or equal to n, and strictly larger than s.
j	The depth of the recursion on hand.
k	The indexing of the node location, from left to right, in the *complete* binary tree.
verbose	Set to true if you'd like to see algorithm progression.
return.env	Set to true if you'd like to get the environment containing the algorithm output, instead of a list.

`binseg.by.thresh.inner`

Inner function for binary segmentation with fixed threshold. The wrapper `binseg.by.thresh()` *is intended to be used by user. Note, when* `thresh` *is set to zero, then this can be used to collect the unbalanced haar wavelet basis.*

Description

Inner function for binary segmentation with fixed threshold. The wrapper `binseg.by.thresh()` is intended to be used by user. Note, when `thresh` is set to zero, then this can be used to collect the unbalanced haar wavelet basis.

Usage

```
binseg.by.thresh.inner(y, thresh, s = 1, e = length(y), j = 0, k = 1,
  verbose = F, env = NULL)
```

Arguments

y	The original data.
thresh	Threshold for $\tilde{X}$. This serves as a stopping rule for the recursion.
s	Starting index, in the vector-valued data. Must be an integer larger than or equal to 1, and strictly smaller than e.
e	Ending index, in the vector-valued data. Must be an integer smaller than or equal to n, and strictly larger than s.
j	The depth of the recursion on hand.
k	The indexing of the node location, from left to right, in the *complete* binary tree.
n	The length of the data y.

---

check.orth.basis *Function to check if the columns in* basislist *are orthonormal.*

Description

Function to check if the columns in basislist are orthonormal.

Usage

```
check.orth.basis(basislist, tol = 1e-10)
```

Arguments

tol	Numerical allowance; letting inner products to be up to size tol.

---

collapse *Helper to collapse matrix (with NAs) to a vector of unique elements.*

Description

Helper to collapse matrix (with NAs) to a vector of unique elements.

Usage

```
collapse(mat)
```

`cplist` *Constructor for cplist object.*

Description

Constructor for cplist object.

Usage

```
cplist(nrow)
```

Arguments

`nrow`	creates an all-NA matrix of dimension nrow x 3. The first two columns must be the numeric (no check yet), but the last column can be of any type you want. Initializes to numeric.

`cusum` *Computes the CUSUM (cumulative sum) statistic. Note, we calculate this as the right-to-left difference, by default.*

Description

Computes the CUSUM (cumulative sum) statistic. Note, we calculate this as the right-to-left difference, by default.

Usage

```
cusum(s, b, e, y, right.to.left = TRUE, contrast.vec = FALSE)
```

Arguments

`s`	starting index.
`b`	breakpoint index.
`e`	end index.
`y`	data.
`right.to.left`	Whether you want right-to-left difference in the cusum calculation. Defaults to TRUE.
`contrast.vec`	If TRUE, then the contrast vector v for cusum=v'y is returned.

`exists.cplist` *gets the value corresponding to the (j,k)'th entry of cplist*

Description

gets the value corresponding to the (j,k)'th entry of cplist

Usage

```
## S3 method for class 'cplist'
exists(cplist, j, k)
```

`extract.cplist` *gets the value corresponding to the (j,k)'th entry of cplist*

Description

gets the value corresponding to the (j,k)'th entry of cplist

Usage

```
## S3 method for class 'cplist'
extract(cplist, j, k)
```

`get.closest` *Helper to get index of closest element of* `val` *out of vector* `allval`

Description

Helper to get index of closest element of `val` out of vector `allval`

Usage

```
get.closest(val, allval)
```

`get.intervals` *Gets ni random intervals whose intervals are sampled from 1:n.*

Description

Gets ni random intervals whose intervals are sampled from 1:n.

Usage

```
get.intervals(n, ni)
```

`get.means` *Gets underlying means for changepoints in y. Wrote this because plot.sbs function doesn't work properly. Example doesn't work now, but here it is: EXAMPLES/get.mean.example.R*

Description

Gets underlying means for changepoints in y. Wrote this because plot.sbs function doesn't work properly. Example doesn't work now, but here it is: EXAMPLES/get.mean.example.R

Usage

```
get.means(y, changepoints, ...)
```

`get.polyhedron` *Function to collect polyhedrons given some output from the binary segmentation function. Takes as input list contatining *list objects.*

Description

Function to collect polyhedrons given some output from the binary segmentation function. Takes as input list contatining *list objects.

Usage

```
get.polyhedron(binseg.results, thresh, verbose = F)
```

`getcusums` *Get all cusums, given S and E*

Description

Get all cusums, given S and E

Usage

```
getcusums(s, e, y)
```

`haarbasis` *Calculates the unbalanced Haar basis for subvector of* `y` *or the vector* `a` *in linear inequalities a*y>=0*

Description

Calculates the unbalanced Haar basis for subvector of `y` or the vector `a` in linear inequalities a*y>=0

Usage

```
haarbasis(s, b, e, n, y, type = c("basis", "ineq"))
```

Arguments

`type`	`basis` to return the unbalanced haar basis, and `ineq` to return linear inequality vector.

Examples

```
y = c(rnorm(10,0,1), rnorm(10,4,1))

# Calculate Haar basis, s:e vs (b+1):e
##mybasis = haarbasis(s = 0, b = 10, e = 20, n = 20, y = y, type = "basis")

# Calculate linear inequality vectors
##myineqs = haarbasis(s = 0, b = 10, e = 20, n = 20, y = y, type = "ineq")
```

`halfspaces` *Calculates the halfspace vectors for the maximizing breakpoint and all the signs.*

Description

Calculates the halfspace vectors for the maximizing breakpoint and all the signs.

Usage

```
halfspaces(s, b, e, thresh, n, y, is.terminal.node = F, verbose = F)
```

Arguments

`is.terminal.node`	T/F for whether the node is one where the threshhold is not breached.

Examples

```
y = c(rnorm(10,0,1), rnorm(10,4,1))
# Calculate linear inequality vectors
##myineqs = halfspaces(s = 0, b = 10, e = 20, n = 20, y = y, type = "ineq")
```

`is.cplist` *Check if object is of class "cplist"*

Description

Check if object is of class "cplist"

Usage

```
is.cplist(someobj)
```

`make.qq.line` *makes QQ line on existing plot.*

Description

makes QQ line on existing plot.

Usage

```
make.qq.line(p, pcol = "red", pch = 16)
```

`make.qqplot.background`
makes QQ plot background without any lines.

Description

makes QQ plot background without any lines.

Usage

```
make.qqplot.background(plot.title = "")
```

`make.qqplot.legend` *makes QQ plot legend on existing plot.*

Description

makes QQ plot legend on existing plot.

Usage

```
make.qqplot.legend(deltas, pch = 16, pcols = rep("red", length(deltas)),
  where.legend = "bottomright")
```

`make.v` *Function to obtain contrast, given the output of binseg(), the SBS algorithm with fixed threshold.*

Description

Function to obtain contrast, given the output of binseg(), the SBS algorithm with fixed threshold.

Usage

```
make.v(test.b, B, Z, n)
```

Arguments

`test.b`	is the location that we want to test.
`B`	vector of breakpoints to be considered
`Z`	signs of B, as
`n`	length of contrast

`make.v.fixed.thresh` *Function to obtain contrast, given the output of binseg(), the SBS algorithm with fixed threshold.*

Description

Function to obtain contrast, given the output of binseg(), the SBS algorithm with fixed threshold.

Usage

```
make.v.fixed.thresh(test.b, bs.output)
```

Arguments

`test.b`	is the location that we want to test.
`bs.output`	list that contains G,u,y,blist,zlist. Manually bundled by user.

`make.v.simple` *Simpler version of make.v with fixed threshold.*

Description

Simpler version of make.v with fixed threshold.

Usage

```
make.v.simple(b, s, e, n, dir)
```

Arguments

`b`	is the location that we want to test.

`mpfr.tnorm.surv` *Returns Prob(Z>z | Z in [a,b]), where mean can be a vector, using multi precision floating point calculations thanks to the Rmpfr package*

Description

Returns Prob(Z>z | Z in [a,b]), where mean can be a vector, using multi precision floating point calculations thanks to the Rmpfr package

Usage

```
mpfr.tnorm.surv(z, mean = 0, sd = 1, a, b, bits = NULL)
```

`mydeparse` *Function to deparse "10:32" to 10:32*

Description

Function to deparse "10:32" to 10:32

Usage

```
mydeparse(mystring)
```

`myprint` *A function I use for debugging.*

Description

A function I use for debugging.

Usage

```
myprint(v1)
```

`perm.t.test` *Conducts a permutation t-test, given two subvectors Example doesn't work now, but here it is: EXAMPLES/perm.t.test.example.R*

Description

Conducts a permutation t-test, given two subvectors Example doesn't work now, but here it is: EXAMPLES/perm.t.test.example.R

Usage

```
perm.t.test(vec1, vec2, nsim = 1000)
```

plot.v *Takes a contrast vector v and optionally B & Z, and plots it*

Description

Takes a contrast vector v and optionally B & Z, and plots it

Usage

```
## S3 method for class 'v'
plot(v, B = NULL, Z = NULL)
```

Arguments

v	Numeric vector containing contrast vector.

print.cplist *Print function*

Description

Print function

Usage

```
## S3 method for class 'cplist'
print(cplist)
```

rid.cplist *Delete (j,k) or jklist,*

Description

Delete (j,k) or jklist,

Usage

```
## S3 method for class 'cplist'
rid(cplist, j = NULL, k = NULL, jklist = NULL)
```

rid.null *Get's rid of null elements in a list*

Description

Get's rid of null elements in a list

Usage

```
## S3 method for class 'null'
rid(mylist)
```

Arguments

mylist	Some list that is suspected to contain some elements equal to NULL.

Examples

```
mylist = list(c(1,1),NULL,c(3,2),NULL,NULL)
mylist = rid.null(mylist)
```

rid_jk_nicely_from_Tcurr *Gets rid of the entries in the terminal node list Tcurr nicely,*

Description

Gets rid of the entries in the terminal node list Tcurr nicely,

Usage

```
rid_jk_nicely_from_Tcurr(Tcurr, Scurr, Ecurr, Tcurr.which.new)
```

Examples

```
Tcurr = list(c(1,1),c(1,2),NULL,NULL)
Ecurr = Scurr = cplist(10)
Ecurr = add(Ecurr, 1,1,7)
Scurr = add(Scurr, 1,1,8)
Ecurr = add(Ecurr, 1,2,11)
Scurr = add(Scurr, 1,2,13)
## rid.jk.nicely(Tcurr,Ecurr,Scurr) ==  list(c(1,2),NULL,NULL,NULL)
```

`sqrt.mn.diff` *Helper function to get right-to-left mean difference, or the contrast vector*

Description

Helper function to get right-to-left mean difference, or the contrast vector

Usage

```
## S3 method for class 'mn.diff'
sqrt(s, b, e, n, y = NA, contrast.vec = FALSE,
  right.to.left = TRUE)
```

`t.statistic` *Calculates a t-statistic for E(v2)-E(v1) given vectors v1 and v2.*

Description

Calculates a t-statistic for E(v2)-E(v1) given vectors v1 and v2.

Usage

```
## S3 method for class 'statistic'
t(v1, v2)
```

`trim` *Function to trim matrices, lists or vectors.*

Description

Function to trim matrices, lists or vectors.

Usage

```
trim(mything, ...)
```

`trim.cplist` *Trim function*

Description

Trim function

Usage

```
trim.cplist(cplist)
```

`trim.list` *Trims a list by deleting the last consecutive elements that are NULL.*

Description

Trims a list by deleting the last consecutive elements that are NULL.

Usage

```
trim.list(mylist, rid.null = FALSE)
```

Arguments

`mylist`	Some list.

`trim.mat` *Function to trim a matrix from the right and bottom, ridding of all-NA rows/columns. Returns NULL if mat is all NA's.*

Description

Function to trim a matrix from the right and bottom, ridding of all-NA rows/columns. Returns NULL if mat is all NA's.

Usage

```
trim.mat(mat, type = c("rowcol", "row"))
```

`trim.vec` *Trims a list by deleting the last consecutive elements that are NULL.*

Description

Trims a list by deleting the last consecutive elements that are NULL.

Usage

```
trim.vec(myvec)
```

Arguments

`mylist`	Some list.

`where_jk.cplist` *Search in cplist$mat for the couplet (j,k) in the first two columns e.g. if j=13 and k = 39, then it searches for the row (13,39,XXX) in an n by 3 matrix.*

Description

Search in cplist$mat for the couplet (j,k) in the first two columns e.g. if j=13 and k = 39, then it searches for the row (13,39,XXX) in an n by 3 matrix.

Usage

```
## S3 method for class 'cplist'
where_jk(cplist, j, k, warn = FALSE)
```

Index